

CLAIMS

1. A method comprising:
segmenting a file into multiple blocks;
computing hashes of each of the blocks to produce corresponding block hash values; and
encrypting the blocks using their corresponding block hash values as encryption keys to produce encrypted blocks.

2. A method as recited in claim 1, wherein the segmenting comprises dividing the file into equal size blocks.

3. A method as recited in claim 1, wherein the encrypting comprises encrypting each block using a symmetric cryptographic cipher and the corresponding block hash value as the symmetric encryption key.

4. A method as recited in claim 1, further comprising storing the encrypted blocks as a primary data stream.

5. A method as recited in claim 4, further comprising storing header information in a separate metadata stream.

6. A method as recited in claim 1, further comprising verifying an authenticity of the encrypted blocks independently of one another.

1 7. A method as recited in claim 1, further comprising modifying content
2 of a block in the file independent of other blocks.

3
4 8. A method as recited in claim 1, further comprising constructing an
5 indexing structure to index individual encrypted blocks.

6
7 9. A method as recited in claim 8, wherein the constructing comprises
8 creating a leaf node for each corresponding encrypted block, the leaf node
9 containing an access value used to decrypt the corresponding encrypted block and
10 a verification value used to verify the corresponding encrypted block.

11
12 10. A method as recited in claim 9, wherein the constructing further
13 comprises hashing an array of the leaf nodes to produce a root.

14
15 11. A method as recited in claim 10, wherein the constructing further
16 comprising digitally signing at least the root.

17
18 12. A method as recited in claim 10, further comprising:
19 storing the root together with header information and per user information;
20 and
21 digitally signing a composite including the root, the header information,
22 and the per user information.

1 **13.** A method as recited in claim 9, wherein the constructing further
2 comprises:

3 grouping leaf nodes into multiple groups;

4 hashing each group of leaf nodes to form intermediate nodes; and

5 hashing an array of the intermediate nodes to produce a root.

6
7 **14.** A method as recited in claim 13, wherein the constructing further
8 comprises digitally signing at least the root.

9
10 **15.** A method as recited in claim 1, further comprising digitally signing
11 at least a portion of the file.

12
13 **16.** A method as recited in claim 1, further comprising generating a
14 delegation certificate that grants other entities permission to collectively
15 authenticate the file in absence of the signature of a last writer to the file.

16
17 **17.** A method as recited in claim 1, wherein the file comprises a sparse
18 file.

19
20 **18.** A method as recited in claim 1, further comprising:
21 encrypting the block hash values with one or more access keys; and
22 encrypting the one or more access keys using one or more keys of users
23 who are granted access to the file.

1 **19.** A method as recited in claim 18, wherein the one or more keys of
2 users comprise one or more public keys.

3
4 **20.** A data structure, embodied on a computer-readable medium,
5 produced by the method of claim 1.

6
7 **21.** One or more computer readable media comprising computer-
8 executable instructions that, when executed, perform the method as recited in
9 claim 1.

10
11 **22.** A method comprising:
12 segmenting a file into multiple blocks;
13 computing hashes of each of the blocks to produce corresponding block
14 hash values;
15 encrypting the blocks using their corresponding block hash values as
16 encryption keys to produce encrypted blocks;
17 storing the encrypted blocks as a primary data stream;
18 creating an indexing structure to index individual encrypted blocks, the
19 indexing structure containing a leaf node for each corresponding encrypted block,
20 the leaf node containing an access value formed by encrypting the block hash
21 value for the corresponding encrypted block using an access key and a verification
22 value formed by hashing the corresponding encrypted block;
23 storing the indexing structure in a separate metadata stream; and
24 encrypting the access key using a public key of a user who is granted access
25 to the file.

1
2 **23.** A method as recited in claim 22, wherein the segmenting comprises
3 dividing the file into equal size blocks.

4
5 **24.** A method as recited in claim 22, wherein the encrypting of the
6 blocks comprises encrypting each block using a symmetric cryptographic cipher
7 and the corresponding block hash value as the symmetric encryption key.

8
9 **25.** A method as recited in claim 22, further comprising verifying an
10 authenticity of a target encrypted block independently of other encrypted blocks
11 by traversing the indexing structure to a leaf node associated with the target
12 encrypted block and using the verification value in the leaf node associated with
13 the target encrypted block.

14
15 **26.** A method as recited in claim 22, further comprising:
16 traversing the indexing structure to a leaf node associated with a target
17 block;
18 decrypting the target block using the access value of the leaf node
19 associated with the target block; and
20 reading the target block following said decrypting.

21
22 **27.** A method as recited in claim 26, further comprising:
23 modifying the target block of the file to produce a modified target block;
24 computing a hash value of the modified target block;
25

1 encrypting the modified target block using the hash value as an encryption
2 key to produce a modified encrypted block; and
3 recreating a new leaf node for the modified encrypted block.
4

5 **28.** A method as recited in claim 22, wherein the creating further
6 comprises:

7 grouping leaf nodes into multiple groups;
8 hashing each group of leaf nodes to form intermediate nodes of the
9 indexing structure; and
10 hashing an array of the intermediate nodes to produce a root.
11

12 **29.** A method as recited in claim 28, wherein the constructing further
13 comprises digitally signing at least the root.
14

15 **30.** A method as recited in claim 22, further comprising digitally signing
16 at least a portion of the metadata stream.
17

18 **31.** A method as recited in claim 22, further comprising generating a
19 delegation certificate that grants other entities permission to collectively
20 authenticate the file in absence of the signature of a last writer to the file.
21
22
23
24
25

1 **32.** A method as recited in claim 22, wherein the file comprises a sparse
2 file in which at least one of the blocks contains no data, the method further
3 comprising:

4 differentiating non-data blocks of the sparse file that contain no substantive
5 content from the data blocks of the sparse file that contain substantive data; and

6 deallocating portions of the metadata stream that pertain to the non-data
7 blocks in the data stream.

8
9 **33.** A data structure, embodied on a computer-readable medium,
10 produced by the method of claim 22.

11
12 **34.** One or more computer readable media comprising computer-
13 executable instructions that, when executed, perform the method as recited in
14 claim 22.

15
16 **35.** A method comprising:

17 creating a primary data stream containing an encrypted file that is encrypted
18 using at least one hash of some contents of the file; and

19 creating a metadata stream containing information pertaining to the
20 encrypted file, the information including decryption capabilities used to decrypt
21 the file and verification capabilities used to verify portions of the file without
22 access to decryption keys.

1 **36.** A method comprising:

2 accessing a file composed of a data stream and a metadata stream, the data
3 stream containing multiple encrypted blocks that are each encrypted using hashes
4 of a plaintext version of the encrypted blocks, the metadata stream containing an
5 indexing structure to index to the individual encrypted blocks, the indexing
6 structure having a leaf node for each corresponding encrypted block that contains
7 a verification value used to verify the corresponding encrypted block;

8 traversing the indexing structure to a leaf node associated with a target
9 encrypted block; and

10 verifying an authenticity of the target encrypted block independently of
11 other encrypted blocks by using the verification value in the leaf node associated
12 with the target encrypted block.

13
14 **37.** A method as recited in claim 36, wherein the indexing structure
15 contains a root and zero or more intervening nodes between the root and the leaf
16 nodes, the traversing further comprising verifying an authenticity of the root and
17 any intervening nodes on a path from the root to the leaf node associated with the
18 target encrypted block.

19
20 **38.** A method as recited in claim 36, wherein the indexing structure is at
21 least partially digitally signed with a digital signature, the method further
22 comprising evaluating an authenticity of the digital signature.

1 **39.** A method as recited in claim 36, wherein the verification value in
2 the leaf node is a hash value of the target encrypted block, and the verifying
3 comprises:

4 computing a current hash value of the target encrypted block; and
5 comparing the current hash value with the hash value in the leaf node.
6

7 **40.** A method for reading a file stored in a distributed file system, the
8 file containing a data stream with multiple encrypted blocks and a metadata stream
9 with an indexing structure to index the encrypted blocks individually, the indexing
10 structure having a leaf node for each corresponding encrypted block that contains
11 an access value used to decrypt the corresponding encrypted block, the method
12 comprising:

13 indexing into the indexing structure to a leaf node associated with a target
14 encrypted block;

15 decrypting the target encrypted block using the access value of the leaf
16 node associated with the target encrypted block; and

17 reading the target encrypted block following said decrypting.
18

19 **41.** A method as recited in claim 40, wherein the access value in the leaf
20 node is an encrypted version of symmetric key used to encrypt the file block, the
21 symmetric key being generated by hashing the file block.
22
23
24
25

1 **42.** A method for writing to a file stored in a distributed file system, the
2 file containing a data stream with multiple encrypted blocks and a metadata stream
3 with an indexing structure to index to the encrypted blocks individually, the
4 method comprising:

5 modifying a block of the file;
6 computing a hash value of the block;
7 encrypting the block using the hash value as an encryption key to produce
8 an encrypted block; and
9 reconstructing a portion of the indexing structure that references the
10 encrypted block.

11
12 **43.** A method as recited in claim 42, wherein the modifying the block
13 comprises writing data to the block.

14
15 **44.** A method as recited in claim 42, wherein the indexing structure
16 includes a leaf node for each corresponding encrypted block, and the
17 reconstructing comprises creating a new leaf node for the encrypted block, the
18 new leaf node containing an encrypted version of the hash value and a hash of the
19 encrypted block.

20
21 **45.** A method comprising:
22 segmenting a sparse file into multiple blocks;
23 differentiating non-data blocks in the sparse file that contain no substantive
24 content from data blocks in the sparse file that contain substantive data;
25 creating an indexing structure to index individual blocks; and

1 deallocating storage of the non-data blocks and portions of the indexing
2 structure that reference the non-data blocks.

3
4 **46.** A method as recited in claim 45, further comprising storing the data
5 blocks in a data stream and the indexing structure in a metadata stream.

6
7 **47.** A method as recited in claim 45, further comprising:
8 computing a hash of each of the data blocks to produce block hash values;
9 and
10 encrypting the data blocks using their corresponding block hash values as
11 encryption keys to produce encrypted data blocks.

12
13 **48.** A method as recited in claim 47, wherein the indexing structure
14 contains first leaf nodes for each corresponding data blocks and second leaf nodes
15 for each corresponding non-data block, the first leaf nodes containing an access
16 value for use in decrypting the encrypted data blocks and a verification value for
17 use in verifying individual encrypted data block independently of other encrypted
18 data blocks.

19
20 **49.** A data structure, embodied on a computer-readable medium,
21 produced by the method of claim 45.

1 **50.** One or more computer readable media comprising computer-
2 executable instructions that, when executed, perform the method as recited in
3 claim 45.

4
5 **51.** A method comprising:
6 segmenting a sparse file into multiple blocks, the sparse file containing at
7 least one non-data block that contains no substantive data;
8 differentiating the non-data blocks from data blocks of the sparse file that
9 contain substantive data;
10 computing hashes of each of the data blocks to produce block hash values;
11 encrypting the data blocks using their corresponding block hash values as
12 encryption keys to produce encrypted data blocks;
13 creating an indexing structure to index individual blocks, the indexing
14 structure containing first leaf nodes for each corresponding encrypted data block
15 and second leaf nodes for each corresponding non-data block, the first leaf nodes
16 containing an access value formed by encrypting the block hash value for the
17 corresponding encrypted block using an access key and a verification value
18 formed by hashing the corresponding encrypted block; and
19 setting the second leaf nodes to a first binary value.

20
21 **52.** A method as recited in claim 51, further comprising storing the
22 encrypted blocks in a data stream and the indexing structure in a metadata stream.
23
24
25

1 **53.** A method as recited in claim 52, further comprising deallocating
2 portions of the metadata stream that hold the second leaf nodes.

3
4 **54.** A data structure, embodied on a computer-readable medium,
5 produced by the method of claim 51.

6
7 **55.** One or more computer readable media comprising computer-
8 executable instructions that, when executed, perform the method as recited in
9 claim 51.

10
11 **56.** One or more computer readable media comprising computer-
12 executable instructions that, when executed, direct a computing device to:

13 segment a file into multiple blocks;

14 hash each of the blocks to produce block hash values;

15 encrypt the blocks using their corresponding block hash values as
16 encryption keys to produce encrypted blocks;

17 create an indexing structure to index individual encrypted blocks, the
18 indexing structure containing a leaf node for each corresponding encrypted block,
19 the leaf node containing an access value formed by encrypting the block hash
20 value for the corresponding encrypted block using an access key and a verification
21 value formed by hashing the corresponding encrypted block;

22 encrypt the access key using a public key of a user who is granted access to
23 the file.

1 **57.** One or more computer readable media as recited in claim 56, further
2 comprising computer-executable instructions that, when executed, direct a
3 computing device to:

4 store the encrypted blocks as a primary data stream; and
5 store the indexing structure in a separate metadata stream.
6

7 **58.** One or more computer readable media as recited in claim 56, further
8 comprising computer-executable instructions that, when executed, direct a
9 computing device to segment the file into equal size blocks.
10

11 **59.** One or more computer readable media as recited in claim 56,
12 wherein the blocks are encrypted using a symmetric cryptographic cipher and the
13 access key is encrypted using an asymmetric cryptographic cipher.
14

15 **60.** One or more computer readable media as recited in claim 56, further
16 comprising computer-executable instructions that, when executed, direct a
17 computing device to verify an authenticity of a target encrypted block
18 independently of other encrypted blocks by traversing the indexing structure to a
19 leaf node associated with the target encrypted block and using the verification
20 value in the leaf node associated with the target encrypted block.
21
22
23
24
25

1 **61.** A method as recited in claim 60, wherein the indexing structure
2 contains a root and zero or more intervening nodes between the root and the leaf
3 nodes, the traversing further comprising verifying an authenticity of the root and
4 any intervening nodes on a path from the root to the leaf node associated with the
5 target encrypted block.

6
7 **62.** One or more computer readable media as recited in claim 56, further
8 comprising computer-executable instructions that, when executed, direct a
9 computing device to:

10 decrypt a target block using an access value of a leaf node associated with
11 the target block; and
12 read the target block after it is decrypted.

13
14 **63.** One or more computer readable media as recited in claim 62, further
15 comprising computer-executable instructions that, when executed, direct a
16 computing device to:

17 modify the target block to produce a modified target block;
18 hash the modified target block to produce a hash value;
19 encrypt the modified target block using the hash value as an encryption key
20 to produce a modified encrypted block; and
21 recreate a new leaf node for the modified encrypted block.

1 **64.** One or more computer readable media as recited in claim 56, further
2 comprising computer-executable instructions that, when executed, direct a
3 computing device to:

4 group leaf nodes into multiple groups;

5 hash each group of leaf nodes to form intermediate nodes of the indexing
6 structure; and

7 hash an array of the intermediate nodes to produce a root.
8

9 **65.** One or more computer readable media as recited in claim 64, further
10 comprising computer-executable instructions that, when executed, direct a
11 computing device to digitally sign at least the root.
12

13 **66.** One or more computer readable media as recited in claim 56, further
14 comprising computer-executable instructions that, when executed, direct a
15 computing device to digitally sign at least a portion of the metadata stream.
16

17 **67.** One or more computer readable media as recited in claim 56, further
18 comprising computer-executable instructions that, when executed, direct a
19 computing device to generate a delegation certificate that grants other entities
20 permission to collectively authenticate the file in absence of the signature of a last
21 writer to the file.
22
23
24
25

1 **68.** One or more computer readable media as recited in claim 56,
2 wherein the file comprises a sparse file in which at least one of the blocks contains
3 no substantive data, the media further comprising computer-executable
4 instructions that, when executed, direct a computing device to:

5 differentiate non-data blocks of the sparse file that contain no substantive
6 content from the data blocks of the sparse file that contain substantive data; and

7 deallocate portions of the metadata stream that pertain to the non-data
8 blocks in the data stream.

9
10 **69.** In a distributed file system that stores files across multiple
11 computers, wherein each file contains a data stream with multiple encrypted
12 blocks and a metadata stream with an indexing structure to index the encrypted
13 blocks individually, the indexing structure having a leaf node for each
14 corresponding encrypted block that contains a verification value used to verify the
15 corresponding encrypted block, one or more computer readable media comprising
16 computer-executable instructions that, when executed, direct a computing device
17 to:

18 traverse the indexing structure to a leaf node associated with a target
19 encrypted block; and

20 verify an authenticity of the target encrypted block independently of other
21 encrypted blocks by using the verification value in the leaf node associated with
22 the target encrypted block.

1 **70.** One or more computer readable media as recited in claim 69,
2 wherein the indexing structure contains a root and zero or more intervening nodes
3 between the root and the leaf nodes, further comprising computer-executable
4 instructions that, when executed, direct a computing device to verify an
5 authenticity of the root and any intervening nodes on a path from the root to the
6 leaf node associated with the target encrypted block.

7
8 **71.** In a distributed file system that stores files across multiple
9 computers, wherein each file contains a data stream with multiple encrypted
10 blocks and a metadata stream with an indexing structure to index the encrypted
11 blocks individually, the indexing structure having a leaf node for each
12 corresponding encrypted block that contains an access value used to decrypt the
13 corresponding encrypted block, one or more computer readable media comprising
14 computer-executable instructions that, when executed, direct a computing device
15 to:

16 index into the indexing structure to a leaf node associated with a target
17 encrypted block;

18 decrypt the target encrypted block using the access value of the leaf node
19 associated with the target encrypted block; and

20 read the target encrypted block following said decrypting.

21
22
23
24
25

1 72. In a distributed file system that stores files across multiple
2 computers, the file containing a data stream with multiple encrypted blocks and a
3 metadata stream with an indexing structure to index to the encrypted blocks
4 individually, one or more computer readable media comprising computer-
5 executable instructions that, when executed, direct a computing device to:

6 modify a block of the file;

7 compute a hash value of the block;

8 encrypt the block using the hash value as an encryption key to produce an
9 encrypted block; and

10 reconstruct a portion of the indexing structure that references the encrypted
11 block.

12
13 73. One or more computer readable media comprising computer-
14 executable instructions that, when executed, direct a computing device to:

15 segment a sparse file into multiple blocks, the sparse file containing at least
16 one non-data block that contains no substantive data;

17 differentiate the non-data blocks from data blocks of the sparse file that
18 contain substantive data;

19 compute hashes of each of the data blocks to produce block hash values;

20 encrypt the data blocks using their corresponding block hash values as
21 encryption keys to produce encrypted data blocks;

22 creating an indexing structure to index the non-data blocks and the
23 encrypted data blocks; and

24 deallocate portions of the indexing structure that reference the non-data
25 blocks.

1
2 **74.** A distributed file system comprising:
3 a client component resident at a first computer to facilitate creation of a file
4 by segmenting the file into multiple blocks and encrypting each block using its
5 own hash value as an encryption key; and

6 a server component resident at a second computer to store the encrypted
7 file.
8

9 **75.** A distributed file system as recited in claim 74, wherein the client
10 component divides the file into equal size blocks.
11

12 **76.** A distributed file system as recited in claim 74, wherein the
13 encrypted blocks are stored as a primary data stream.
14

15 **77.** A distributed file system as recited in claim 76, wherein the client
16 component creates header information that is stored as a separate metadata stream.
17

18 **78.** A distributed file system as recited in claim 74, wherein the client
19 component verifies an authenticity of the encrypted blocks independently of one
20 another.
21

22 **79.** A distributed file system as recited in claim 74, wherein the client
23 component modifies content of a block in the file independent of other blocks.
24
25

1 **80.** A distributed file system as recited in claim 74, wherein the client
2 component digitally signs the file.

3
4 **81.** A component in a distributed file system in which file are stored
5 across multiple distributed computers, the component comprising:

6 a segmenting module to divide a file into multiple blocks;
7 a hash module to hash each of the blocks to produce block hash values;
8 a cryptographic engine to encrypt the blocks using their corresponding
9 block hash values as encryption keys to produce encrypted blocks; and
10 an index builder to create an indexing structure for indexing individual
11 encrypted blocks, the indexing structure containing a leaf node for each
12 corresponding encrypted block, the leaf node containing an access value formed
13 by encrypting the block hash value for the corresponding encrypted block using an
14 access key and a verification value formed by hashing the corresponding
15 encrypted block.

16
17 **82.** A component as recited in claim 81, wherein the cryptographic
18 engine is further configured to encrypt the access key using a key of a user who is
19 granted access to the file.

20
21 **83.** A component as recited in claim 81, wherein the segmenting module
22 divides the file into equal size blocks.

1 **84.** A component as recited in claim 81, wherein cryptographic engine
2 employs a symmetric cryptographic cipher to encrypt the blocks.

3
4 **85.** A component as recited in claim 81, further comprising a
5 verification module to verify an authenticity of a target encrypted block
6 independently of other encrypted blocks by traversing the indexing structure to a
7 leaf node associated with the target encrypted block and using the verification
8 value in the leaf node associated with the target encrypted block.

9
10 **86.** A component as recited in claim 85, wherein the indexing structure
11 contains a root and zero or more intervening nodes between the root and the leaf
12 nodes, the verification module being configured to verify an authenticity of the
13 root and any intervening nodes on a path from the root to the leaf node associated
14 with the target encrypted block.

15
16 **87.** A component as recited in claim 81, further comprising a control
17 module to index into the indexing structure to a leaf node associated with a target
18 block, decrypt the target block using the access value of the leaf node associated
19 with the target block, and read the target block.

20
21 **88.** A component as recited in claim 87, where upon modification of the
22 target block:

23 the hash module hashes the modified target block to produce a new hash
24 value;

1 the cryptographic engine encrypts the modified target block using the new
2 hash value as an encryption key to produce a modified encrypted block; and
3 the index builder creates a new leaf node for the modified encrypted block.
4

5 **89.** A component as recited in claim 81, wherein the index builder is
6 configured to create intermediate nodes that index the leaf nodes.
7

8 **90.** A component as recited in claim 81, further comprising a signing
9 module to digitally sign at least a portion of the indexing structure.
10

11 **91.** A component in a distributed file system in which files are stored
12 across multiple distributed computers, the component comprising:
13

14 a segmenting module to divide a sparse file into multiple blocks, the sparse
15 file containing at least one non-data block that contains no substantive data;
16

17 a control module to differentiate the non-data blocks from data blocks of
18 the sparse file that contain substantive data;
19

20 a hash module to hash each of the data blocks to produce block hash values;
21

22 a cryptographic engine to encrypt the data blocks using their corresponding
23 block hash values as encryption keys to produce encrypted blocks; and
24

25 an index builder to create an indexing structure to index individual blocks,
the indexing structure containing first leaf nodes for each corresponding encrypted
block and second leaf nodes for each corresponding non-data block, the first leaf
nodes containing an access value formed by encrypting the block hash value for
the corresponding encrypted block using an access key and a verification value

1 formed by hashing the corresponding encrypted block, the second leaf nodes being
2 set to a first binary value.

3
4 **92.** A component as recited in claim 91, wherein the encrypted blocks
5 are stored in a data stream and the indexing structure is stored in a metadata
6 stream.

7
8 **93.** A component as recited in claim 92, wherein the control module
9 deallocates portions of the metadata stream that hold the second leaf nodes.

10
11 **94.** A distributed file system comprising:
12 means for creating a primary data stream containing an encrypted file that
13 is encrypted using at least one hash of some contents of the file; and
14 means for creating a metadata stream containing information pertaining to
15 the encrypted file, the information including decryption capabilities used to
16 decrypt the file and verification capabilities used to verify the file.

17
18 **95.** A distributed file system as recited in claim 94, wherein the file in
19 the primary stream comprises multiple encrypted blocks, where each block is
20 hashed and encrypted using a hash value of the block as an encryption key.

1 **96.** A distributed file system as recited in claim 94, wherein the means
2 for creating the metadata stream comprises means for constructing leaf nodes for
3 corresponding blocks, the leaf nodes containing an access value used to decrypt
4 the corresponding encrypted block and a verification value used to verify the
5 corresponding encrypted block.

6
7 **97.** A data structure stored on a computer-readable medium,
8 comprising:

9 a primary data stream containing an encrypted file composed of multiple
10 encrypted blocks, each block being separately encrypted by a symmetric cipher
11 that uses a hash of the block as an encryption key;

12 a metadata stream containing information pertaining to the encrypted file.

13
14 **98.** A data structure as recited in claim 97, wherein information includes
15 decryption capabilities used to decrypt the file.

16
17 **99.** A data structure as recited in claim 97, wherein information includes
18 verification capabilities used to verify the file.

19
20 **100.** A data structure as recited in claim 97, wherein the metadata
21 stream comprises:

22 a header with one or more fields that describe the encrypted file; and

23 an indexing structure to index individual encrypted blocks.
24
25

1 **101.** A data structure stored on a computer-readable medium,
2 comprising:

3 multiple encrypted file blocks, each encrypted file block being encrypted by
4 a symmetric cipher that uses a hash of the block as an encryption key; and
5 an indexing structure to index individual encrypted file blocks
6 independently of other encrypted file blocks.

7
8 **102.** A data structure as recited in claim 101, wherein the indexing
9 structure comprises a leaf node for each corresponding encrypted block, the leaf
10 node containing an access value formed by encrypting the hash of the block using
11 a randomly generated key and a verification value formed by hashing the
12 corresponding encrypted block.

13
14 **103.** A data structure as recited in claim 102, further comprising a user
15 key list containing one or more identities of user who have access to the encrypted
16 file blocks, each identity including an entry with an encrypted version of the
17 randomly generated key that is encrypted using the user's public key.

18
19 **104.** A data structure as recited in claim 101, wherein the indexing
20 structure comprises:

21 a leaf node for each corresponding encrypted block, the leaf node
22 containing an access value formed by encrypting the hash of the block using a
23 randomly generated key and a verification value formed by hashing the
24 corresponding encrypted block; and

25 a root node formed by hashing an array of the leaf nodes.

1
2 **105.** A data structure as recited in claim 104, wherein the indexing
3 structure further comprises a digital signature produced by digitally signing at
4 least the root node.
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25